

Teoria Algorítmica dos Jogos - 2009.2
Lista de exercícios 1

1. Considere o seguinte jogo de 2 jogadores cujas utilidades são definidas abaixo. Se a posição (i, j) contém (a, b) , então $u_1(i, j) = a$ e $u_2(i, j) = b$. Ou seja, se os Jogadores 1 e 2 fizerem as jogadas i e j , respectivamente, então as utilidades serão a e b .

$$\begin{bmatrix} (7, 5) & (9, 2) \\ (8, 3) & (3, 6) \end{bmatrix}$$

- (a) Considere as estratégias mistas $x_1 = (1/4, 3/4)$ e $x_2 = (1/3, 2/3)$ dos Jogadores 1 e 2 respectivamente. Qual é a utilidade esperada de cada jogador?
- (b) Existe um equilíbrio puro de Nash nesse jogo? Se existe, mostre-o. Senão, explique porque não existe. Em qualquer caso, apresente um equilíbrio de Nash.

2. Considere o seguinte jogo de soma zero com 2 jogadores, representando as utilidades do Jogador 1. Obtenha computacionalmente o equilíbrio de Nash desse jogo usando o Octave (obtenha as probabilidades de cada jogador bem como sua utilidade esperada).

$$A = \begin{bmatrix} 3 & 2 & 5 & 2 & 5 \\ 4 & 1 & 5 & 3 & 2 \\ 1 & 4 & 8 & 1 & 7 \\ 6 & 1 & 3 & 5 & 6 \end{bmatrix}$$

3. Faça o mesmo da questão anterior agora para jogos aleatórios. Gere aleatoriamente números inteiros n_1 e n_2 entre 3 e 10. Gere aleatoriamente uma matriz A com n_1 linhas e n_2 colunas, com números entre -10 e 10 com três casas decimais. Tendo a matriz A como as utilidades do Jogador 1 em um jogo de soma zero com 2 jogadores, obtenha computacionalmente o equilíbrio de Nash usando o Octave. Depois disso, tente agora fazer com que a matriz A seja simétrica e observe os valores das utilidades obtidos no equilíbrio de Nash.

4. Sobre jogos de soma zero com 2 jogadores, responda o seguinte:

- (a) Mostre que, em todo jogo simétrico de soma zero com 2 jogadores, o valor do minimax é zero.
- (b) Construa um jogo de soma zero com 2 jogadores onde, se o Jogador 2 jogar totalmente aleatório, vale mais a pena para o Jogador 1 jogar diferente do minimax. É proibido usar o mesmo exemplo do livro “Algorithms” de Dasgupta et al.

5. Ao descrever o algoritmo de Lemke-Howson, nós usamos o fato trivial de que em um grafo finito não-direcionado, onde cada vértice possui grau 1 ou 2, se existe um vértice s de grau 1, então existe um outro vértice $t \neq s$ de grau 1 na mesma componente conexa de s . Prove esse fato simples.

6. Na Seção 2.1.1 do livro “Algorithmic Game Theory”, é explicado como obter os valores das probabilidades em um equilíbrio de Nash dado o seu suporte. No caso de um jogo com 2 jogadores, é dito que o sistema de equações a ser resolvido é um sistema linear. Explique este fato simples. Mostre o sistema de equações a ser resolvido para o exemplo abaixo, onde A e $B = A^T$ são as utilidades dos Jogadores 1 e 2, sabendo que o

suporte do Jogador 1 são as jogadas 2 e 3 e o suporte do Jogador 2 também são as jogadas 2 e 3. Ou seja, esse exemplo é o mesmo jogo simétrico da seção 2.1.1. Resolva este sistema.

$$A = \begin{bmatrix} 0 & 3 & 0 \\ 0 & 0 & 3 \\ 2 & 2 & 2 \end{bmatrix}$$

7. Nessa questão você vai implementar um algoritmo que faz o mesmo da questão anterior agora para jogos aleatórios. Gere aleatoriamente números inteiros n_1 e n_2 entre 3 e 10. Gere aleatoriamente uma matriz A n_1 por n_2 e uma matriz B n_2 por n_1 , com números entre -10 e 10 com três casas decimais. As matrizes A e B serão respectivamente as utilidades dos Jogadores 1 e 2 em um jogo com 2 jogadores. Faça um algoritmo que receba como entrada as matrizes A e B e um candidato a suporte para cada jogador. O seu algoritmo deve obter computacionalmente o sistema de equações lineares, resolvê-lo usando o Octave e responder se tais suportes formam de fato um equilíbrio de Nash.

8. Nessa questão você vai implementar um outro algoritmo que usa o anterior como sub-rotina. Você deve gerar todos os suportes possíveis, primeiro os de tamanho 1 para ambos os jogadores, depois os de tamanho 2 para ambos os jogadores e assim sucessivamente, até encontrar um equilíbrio de Nash. Ou seja, o algoritmo é exponencial. Aqui estamos supondo que os suportes dos jogadores terão o mesmo tamanho, que é geralmente o caso em jogos aleatórios. Rode seu algoritmo muitas vezes e nos diga o tempo médio de execução.